PRÁCTICA 4
PROBLEMA DEL CABALLO (KNIGHT'S TOUR)

Algorítmica

Grado en Ingeniería Informática
Universidad de granada

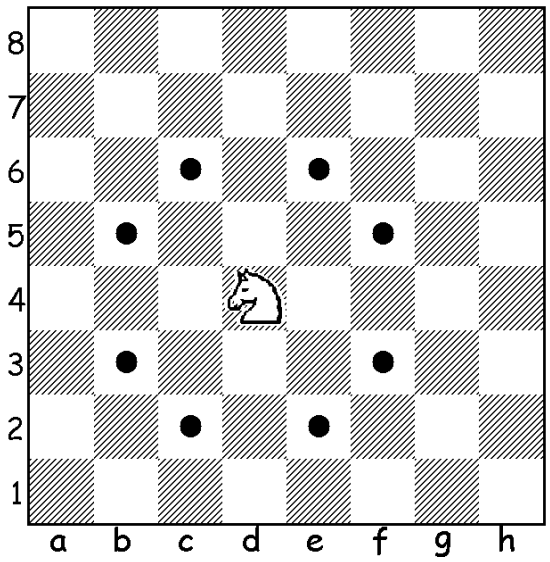
ADRA SANCHEZ RUIZ
CRISTINA GARRIDO AMADOR
JUAN MANUEL CASTILLO NIEVAS
LUIS LIÑÁN VILLAFRANCA
MARIANA ORIHUELA CAZORLA GRUPO A2

17 de mayo de 2017, Granada

Índice

1. Presentación del problema

Consideremos un caballo de ajedrez colocado en una posición, (x_0, y_0), de un tablero de dimensiones nxn. Se trata de encontrar, si es posible, una sucesión de movimientos válidos del caballo de forma que este pueda visitar todas y cada una de las casillas del tablero sin repetir ninguna.



Problema: diseñar e implementar un algoritmo de vuelta atrás que encuentre una serie de movimientos que resuelva el problema, o indique que no es posible.

2. Pseudocódigo del algoritmo backtracking

Algorithm 1 Algoritmo Backtracking

```
1: function BACK_RECURSIVO(solucion, k)
2:     if solucion.SolucionEncontrada() then
3:         if soluciones_totales < 1 then
4:             solucion.MostrarSolucion()
5:         soluciones_totales++
6:     else
7:         solucion.IniciarComponente(k)
8:         solucion.SigValComp(k)
9:         while !solucion.TodosGenerados(k) do
10:            if solucion.Factible(k) then
11:                back_recursivo(solucion, k + 1)
12:                solucion.VueltaAtras()
13:            solucion.SigValComp(k)
```

Las soluciones utilizadas auxiliarmente en el algoritmo de Backtracking son las siguientes:

- SolucionEncontrada():
  Comprueba todas las posiciones del tablero para ver si se ha pasado por cada una de ellas.

- MostrarSolucion():
  Escribe por pantalla la secuencia de movimientos con la que ha conseguido la solución.

- IniciarComponente(k):
  Inicializa una componente del vector solución a 0 ("begin" en el dominio de cada componente de la solución).

- SigValComp(k):
  Selecciona el siguiente valor válido de nuestro dominio.
  Dominio = 0,1,2,3,4,5,6,7,8,9
  $0 \rightarrow$ begin
  $9 \rightarrow$ end

- TodosGenerados(k):
  Comprueba que en la componente k del vector solución se han generado todos los posibles movimientos (si ha llegado la componente al "end"del dominio)

- Factible(k):
  Comprueba si el movimiento actual en la posición k de nuestro vector solución es factible, es decir, si se puede realizar sobre el tablero.

- VueltaAtras():
  Pone a no visitada la posición del tablero actual, y borra la siguiente componente de la solución..

3. Salida del algoritmo

Ejecutando el programa con un tamaño de tablero de 5x5 y la posición inicial de (0, 0):

```
Solucion:
Accion 1; Accion 1; Accion 3; Accion 5; Accion 6; Accion 1; Accion 7; Accion 4;
     Accion 3; Accion 1; Accion 8; Accion 6; Accion 5; Accion 3; Accion 2;
    Accion 8; Accion 7; Accion 5; Accion 3; Accion 3; Accion 8; Accion 1;
    Accion 6; Accion 7;

Se han encontrado en total 304 soluciones en 0.165817 segundos.
```

Ejecutando el programa con un tamaño de tablero de 5x5 y la posición inicial de (2, 2):

```
Solucion:
Accion 1; Accion 4; Accion 5; Accion 7; Accion 8; Accion 2; Accion 3; Accion 5;
     Accion 6; Accion 8; Accion 1; Accion 3; Accion 4; Accion 6; Accion 7;
    Accion 1; Accion 3; Accion 1; Accion 6; Accion 7; Accion 4; Accion 5;
    Accion 2; Accion 3;

Se han encontrado en total 64 soluciones en 0.0584977 segundos.
```

Ejecutando el programa con un tamaño de tablero de 6x6 y la posición inicial de (0, 0):

```
Solucion:
Accion 1; Accion 1; Accion 2; Accion 4; Accion 5; Accion 6; Accion 1; Accion 1;
     Accion 4; Accion 5; Accion 7; Accion 2; Accion 4; Accion 7; Accion 6;
    Accion 8; Accion 1; Accion 2; Accion 3; Accion 6; Accion 8; Accion 6;
    Accion 3; Accion 5; Accion 3; Accion 2; Accion 8; Accion 1; Accion 6;
    Accion 7; Accion 5; Accion 3; Accion 6; Accion 1; Accion 8;

Se han encontrado en total 524486 soluciones en 1441.06 segundos.
```

Como podemos observar, dependiendo del punto de inicio, obtendremos distintos número de soluciones y soluciones.

4. Curiosidades y factibilidad del problema

- Conrad demostró en 1994 que siempre existía solución cuando $n >= 5$ en un cuadrado de nxn

- Euler construyó un cuadrado mágico donde las filas y las columnas sumaban 260. El caballo se desplaza desde la casilla 1 hasta la 64 en orden numérico.

1	48	31	50	33	16	63	18
30	51	46	3	62	19	14	35
47	2	49	32	15	34	17	64
52	29	4	45	20	61	36	13
5	44	25	56	9	40	21	60
28	53	8	41	24	57	12	37
43	6	55	26	39	10	59	22
54	27	42	7	58	23	38	11

- No existe consenso acerca del número total de soluciones posibles. Pero, gracias a la ayuda de los ordenadores, en 1995 Löbbing y Wegener pusieron a trabajar a 20 ordenadores para calcular posibles variantes para el paseo del caballo sin repetir ninguna casilla y obtuvieron una cifra de más de 33 billones de soluciones posibles.